

RD-A184 756

MULTI-CHANNEL COMMUNICATIONS SYSTEM ANALYZER(U)

1/1

STANFORD TELECOMMUNICATIONS INC RESTON VA

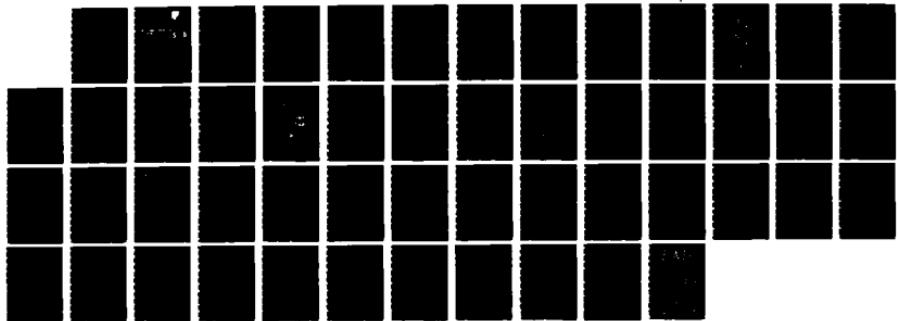
R J KLEINHENZ JUN 87 STI/E-TR-870109 RADC-TR-87-74

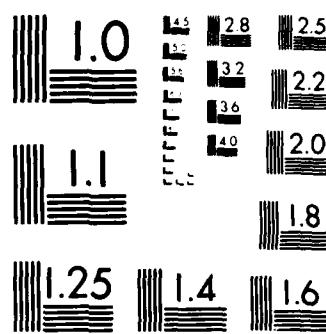
UNCLASSIFIED

F30602-85-C-0089

F/G 25/5

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1964

DTIC FILE COPY

AD-A184 756

RADC-TR-87-74
Final Technical Report
June 1987



12

MULTI-CHANNEL COMMUNICATIONS SYSTEM ANALYZER

DTIC
SELECTED
SEP 16 1987
S D

Stanford Telecommunications, Inc.

Robert J. Kleinhenz

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700**

7 09 15 11 1

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-87-74 has been reviewed and is approved for publication.

APPROVED: *Chuck Y. Wong*
CHUCK Y. WONG
Project Engineer

APPROVED: *B. BEEK*
BRUNO BEEK
Technical Director
Directorate of Communications

FOR THE COMMANDER:

Richard W. Pouliot

RICHARD W. POULIOT
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DCLD) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

11-1-766
Form Approved
OMB No. 0704-0188

REPORT DOCUMENTATION PAGE			
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2a. SECURITY CLASSIFICATION AUTHORITY N/A	2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR870109	
6a. NAME OF PERFORMING ORGANIZATION Stanford Telecommunications, Inc.	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (DCLD)	
6c. ADDRESS (City, State, and ZIP Code) 1761 Business Center Drive Reston VA 22090	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center	
8b. OFFICE SYMBOL (if applicable) DCLD	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-85-C-0089	10. SOURCE OF FUNDING NUMBERS	
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	PROGRAM ELEMENT NO. 62702F	PROJECT NO. 4519	TASK NO. 20
11. TITLE (Include Security Classification) MULTI-CHANNEL COMMUNICATIONS SYSTEM ANALYZER			
12. PERSONAL AUTHOR(S) Robert J. Kleinhenz			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM Aug 85 TO Jan 87	14. DATE OF REPORT (Year, Month, Day) June 1987	15. PAGE COUNT 52
16. SUPPLEMENTARY NOTATION N/A			
17. COSATI CODES	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Communications Test Bed Controller Multiple Channels Statistical Analyzer		
FIELD 25	GROUP	SUB-GROUP	
14	02		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The Multi-Channel Communications System Analyzer (MCCSA) is a computer controller for a general communications test bed. The MCCSA consists of two or more PC AT style micro-computers, connected by a Local Area Network (LAN), off-the-shelf interfaces that permit connections to the test bed, a custom component called the High Speed Serial Data Comparator (HSSDC) to handle data channels up to 50 Mbps, and a combination of custom and off-the-shelf software that controls the experiment, aids in experiment definitions, and performs post-experiment analyses.			
The MCCSA is capable of injecting 50 Mbps test data streams onto the test bed, capturing two 50 Mbps data streams from the test bed, aligning the two captured streams, comparing them for discrepancies (errors), compressing the results of the comparison, and storing the results on magnetic media for later analyses. The MCCSA also supports a facility that permits it to respond (in real-time) to events happening on the test bed and to record the event (and its time of occurrence) for use in creating the post-experiment chronology (over)			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Chuck Y. Wong	22b. TELEPHONE (Include Area Code) (315) 330-7751	22c. OFFICE SYMBOL RADC (DCLD)	

UNCLASSIFIED

and in analyzing the experiment. The MCCSA response to test bed events is user programmable and can serve as a basis for all types of open-loop/closed-loop experiments.

This document highlights the elements of the design and describes the reasons behind the design choices. Several problems, and the design solutions are discussed at length (synchronization, compression, time tagging) to highlight the difficulties involved in creating the MCCAA. The synchronization and compression algorithms are discussed in some detail as are the decisions behind these algorithm's selections.

Accession For	
1 TIS CRA&I	<input checked="" type="checkbox"/>
2 TAB	<input type="checkbox"/>
3 Unannounced	<input type="checkbox"/>
4 Classification	
Distribution	
Availability Classes	
1st	2nd
3rd	4th
A7	

2-1

QUALITY INSPECTED
INSQ

UNCLASSIFIED

TABLE OF CONTENTS

<u>SECTION</u>		<u>PAGE</u>
1	INTRODUCTION.....	1
1.1	Executive Summary.....	1
1.2	Project Review.....	5
2	DESIGN SUMMARY.....	8
2.1	Test Bed Summary.....	8
2.2	Requirements Summary.....	10
2.3	Operational Summary.....	10
2.3.1	Experiment Definition.....	10
2.3.2	Experiment Phase.....	15
2.3.3	Experiment Analyses.....	16
2.4	Hardware Summary.....	17
2.4.1	Off-The-Shelf-Hardware.....	17
2.4.2	Custom Hardware.....	20
2.4.2.1	The Host Interface.....	20
2.4.2.2	The Experiment Interface.....	22
2.4.2.3	Compare/Compression.....	23
2.4.2.4	Transfer Buffers.....	24
2.4.2.5	Events Engine.....	24
2.4.2.6	Data Generator.....	25
2.4.2.7	Firmware.....	25
2.5	Software Summary.....	26
2.5.1	Off-The-Shelf Software.....	26
2.5.2	Custom Software.....	27
3	DESIGN CONSIDERATIONS.....	30
3.1	Summary and Assumptions.....	30
3.1.1	Hardware.....	32
3.1.2	Software.....	33

TABLE OF CONTENTS (Cont'd)

<u>SECTION</u>		<u>PAGE</u>
4	PROBLEMS AND SOLUTIONS.....	35
4.1	Synchronization.....	35
4.2	Time Stamping.....	36
4.3	Compression.....	37
4.4	Hardware Choice.....	38
4.5	Software Choice.....	39
4.6	Customization.....	39
	REFERENCES.....	41

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Multi-Channel Communications System Analyzer.....	2
2	Monthly Technical Hours.....	6
3	Sample Test Bed.....	9
4	High Level Operations.....	13
5	High Speed Serial Data Comparator.....	21

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
1	Actual Technical Hours by Month and Skill Level.....	7
2	Requirements Summary.....	11
3	Off-The-Shelf Components List.....	19

SECTION 1

INTRODUCTION

This report summarizes the design of the Multi-Channel Communications System Analyzer (MCCSA) accomplished for the Rome Air Development Center (RADC) by Stanford Telecommunications, Inc.. This project produced a design for a communications test bed controller capable of:

- Supplying a test bed with 50 Mbps data,
- Sustaining test bed throughput of 50 Mbps,
- Compressing the output data to a reasonable size,
- Controlling ancillary devices on a test bed,
- Showing correlations of the data with test bed events, allowing experiments using multiple communications interfaces,
- Allowing experiments that have multiple channels, and
- Permitting control of the hardware through easy-to-use software.

The manner in which these features were produced and the reasons behind the design decisions is the subject of this report.

1.1 EXECUTIVE SUMMARY

Figure 1 gives an overall view of the MCCSA. The first and foremost design consideration of this project was to exploit the wide variety of off-the-shelf peripherals that are available for the IBM PC family of microcomputers. This was accomplished with the addition of only one custom component -- the High Speed Serial Data Comparator (HSSDC). The need for this custom component

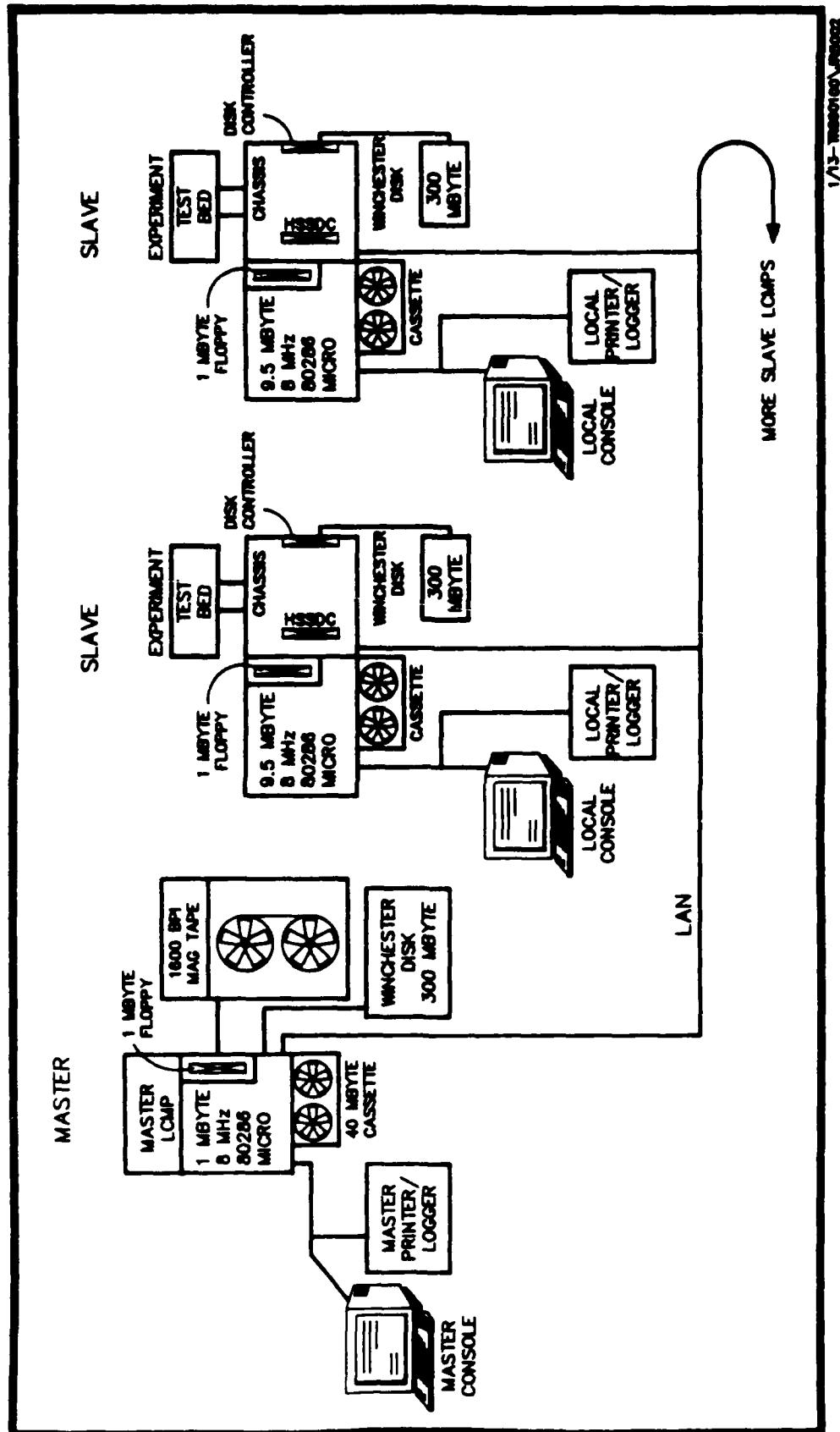


Figure 1. Multi-Channel Communications System Analyzer.

1/13- MARCH 1981 JES/002

arises from the desire to drive the communications test bed at a top data rate of 50 Mbps, and the limited bus bandwidth of the PC family of computers. Typically a PC AT (or equivalent) has a bus bandwidth of approximately 1.5 Mbps spread over 16 data lines for an aggregate bandwidth of about 24 Mbps. To accommodate 50 Mbps data requires a component that will provide data buffering and data compression to squeeze the 50 Mbps data onto the computer's bus.

Another requirement was the desire to recreate an exact error history of the experiment. To do this necessitates the capture and processing of two data streams -- a test stream and a reference stream. A compression algorithm was developed based on a scheme used at RADC in a previous test bed controller. The algorithm records error patterns when they exist but otherwise records the number of error-free sections received since the last error. When the bit error rate is low this scheme is very efficient and can yield tremendous compression. When the error rate is high the scheme, coupled with large buffers, reports the exact error pattern until local on-line storage is exhausted. Typically long duration experiments having bit-error-rates of about .005 or less can be accommodated by the custom hardware without exhausting the local on-line storage. At sustained error rates above .005, experiments may still be done but the risk of exhausting the local storage increases. To preclude this the HSSDC was designed to capture statistics and continue running even if the local data store overflows. In these situations it is not possible to capture exact error data for long periods but meaningful information can still be gained from the statistics capturing facility that is built into the hardware itself.

The MCCSA was also designed to control a wide variety of test bed, digital devices in response to varying test bed conditions. This enables the experimenter to monitor conditions on the test bed for post-experiment analysis. For example, the MCCSA can be set to read a digital power meter every time the instantaneous bit error rate exceeds a prespecified value. In this way the correlation of bit error rate with power at a particular test point can be determined experimentally.

The status of various devices and the occurrence of test bed events can also be recorded for post-experiment correlations. A chronology of events and MCCSA actions can be produced to reveal possible causal relationships between MCCSA gathered data and test bed events. In this way an experimenter can determine the effects of various algorithms by monitoring the relevant variables of the algorithm while changing test bed settings. The MCCSA possesses the ability to carry out actions in response to preprogrammed conditions -- a crucial ability analyzing the dynamics of a particular test bed.

The MCCSA is also able to monitor multiple test bed channels through a Local Area Network (LAN) that links the controlling PCs. Using off-the-shelf software provided with the LAN, each PC can transmit data or status to other PCs on the LAN. This allows the experimenter to analyze complicated, high-speed, communication test beds with a single, integrated controller. In fact, if the aggregate network data rate is less than about 1.5 Mbps, a small network can be analyzed by a single PC. The size of the network that can be analyzed is limited only by the number of PCs allowed on the LAN and the number of peripheral slots available in each PC. Events monitored by one PC can also trigger actions in another PC through the facilities provided by the MCCSA and LAN software, allowing the creation of multiple channel correlations at the conclusion of the experiment.

The analysis done at an experiment's conclusion is provided by a combination of user written software and an off-the-shelf statistical package. The off-the-shelf statistical software gives the user a set of elementary routines that can compute many common statistics of interest (bit error rate, correlations, means, variances, etc.) and allows the user the ability to tailor the experiment data to his own needs. This tailoring process gives the user a way to convert the experiment data to a format acceptable to the user's custom routines. The statistical package also supplies the user with both textual and graphic output and can be the principal means by which a user prints or plots his results.

1.2 PROJECT REVIEW

The first output of this project was a Design Plan [1] for the MCCSA. This document served as a guideline for all work that followed and described the test bed, operations, hardware, and software requirements to be placed on the MCCSA. The Design Plan also suggested methods by which some of the MCCSA design problems would eventually be resolved and established the type of environment that the MCCSA would produce.

After the Design Plan was completed, work commenced on the writing of the two specifications [2,3] and on the preparation of a set of engineering drawings [4]. The first of these two specifications was a three-part B-1 level, hardware specification describing both the off-the-shelf and the custom components of the MCCSA. The second specification was a B-5 level, software specification detailing the off-the-shelf software to be used on the MCCSA and providing a description of the custom software required to drive the MCCSA. The third document, the engineering drawings, showed the functional details of the custom hardware components and provided a design basis for the construction of a prototype.

All of the work for this project was completed on time and within budget. Figure 2 summarizes the effort expended in completing this project. The figure shows a month-by-month history of planned hours and actual hours used to complete the work. Table 1 is also included to show the breakdown of labor by skill categories.

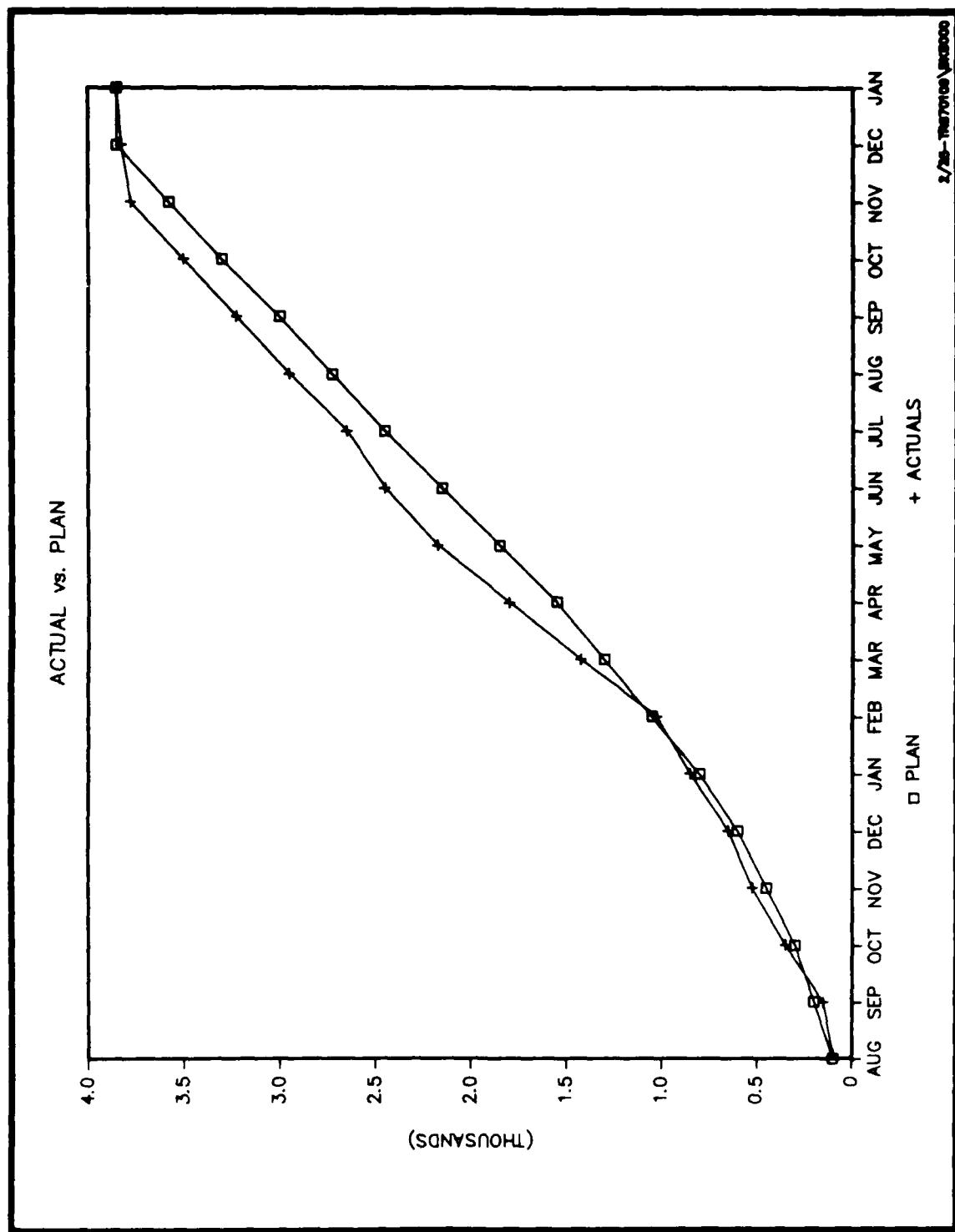


Figure 2. Monthly Technical Hours.

TABLE 1

ACTUAL TECHNICAL HOURS
BY MONTH AND SKILL LEVEL

1986												1987													
JUL AUG SEPT OCT NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEPT OCT NOV DEC JAN TOTALS												JUL AUG SEPT OCT NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEPT OCT NOV DEC JAN TOTALS													
DEPARTMENT	10	16	4	18	2	50	10	16	4	18	2	50	10	16	4	18	2	50	10	16	4	18	2	50	
MANAGER																									
SENIOR ENGINEER SPECIALIST	60	92	74	56	80	16	54	69	68	28	8	605													
ENGINEER SPECIALIST	30	44	64	25	16	89	46	56	50	47	20	21	40	32	40	10	10	10	650						
SENIOR ENGINEER													76	71	52	56	44	80	60	32				471	
ENGINEER 2													44		106	112	150	217	150	36				817	
ENGINEER 1													24	150	228	175	155	38.5	60					2	830.5
CONSULTANT													48	76	84	88	84							32	412
																								3,835.5	

SECTION 2

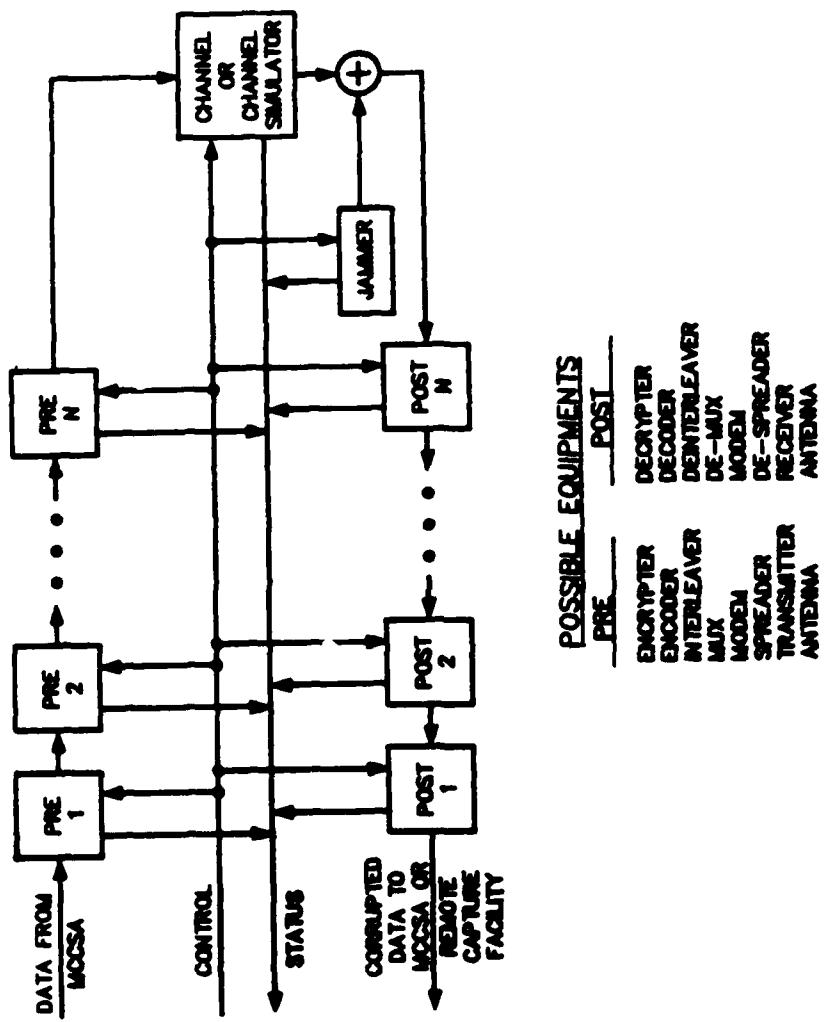
DESIGN SUMMARY

2.1 TEST BED SUMMARY

To accomplish the design of the MCCSA, some basic assumptions were made about the test bed. Initially, it was assumed that all interfaces between the MCCSA and the test bed would be digital. That is, no analog signals would be processed by the MCCSA without first being converted to a digital format. This was not regarded as a major restriction because of the wide variety of analog to digital converters available for the PC family of computers. Moreover, it was assumed that the reading of data and status, and the exercising of control over devices would be accomplished through a set of standard, specified interfaces. Figure 3 gives an example of a one-channel test bed although more complex test beds can be supported.

No other restrictive assumptions were made about the equipment found on the test bed. The specification of interfaces allows a large number of different devices to be connected immediately to the MCCSA and does not preclude the addition of future devices that meet the specification. Since both commercial and military equipment are to be used in the test bed, four commercial interfaces and two military interfaces were chosen for the MCCSA. The interfaces were selected primarily because of the number of different types of equipment they support.

The requirements set forth by RADC included the need to analyze experiments whose data rates may be as large as 50 Mbps or as small as 110 bps. A general purpose interface was required between the test bed and the MCCSA capable of spanning this wide range of data rates. A fiber optics interface was chosen not only because of its ability to meet the data rate requirement but also because of the wide availability of commercial, fiber optic, interface converters.



POSSIBLE EQUIPMENTS

PRE	POST
DECRYPTER	
ENCODER	
DEINTERLEAVER	
DE-MUX	
MODEM	
SPREADER	
TRANSMITTER	
RECEIVER	
ANTENNA	

1/17-1968-100000000000

Figure 3. Sample Test Bed.

It was also assumed that each device on the test bed may have a control/status channel, possibly separate from the data channel, through which the device is commanded and from which the device is monitored. This control/status channel was assumed to meet the requirements of a slow (< 1 Mbps) MCCSA standard interface. This latter assumption is a mild one since control channels are usually slower than the data channels they control.

2.2 REQUIREMENTS SUMMARY

Table 2 gives a functional and performance requirements summary broken down by the categories of Test Bed, Operations, Hardware, and Software. The table entries reflect the desires of RADC as well as the limitations placed on the MCCSA by current technology.

2.3 OPERATIONAL SUMMARY

Since the average user sees the MCCSA primarily from an operational standpoint, it is important that the user interface be easy to understand and to use. Figure 4 shows the high-level operations that occur during MCCSA operations. Every experiment is divided into three phases -- the pre-experiment or definition phase, the experiment phase, and the post-experiment or analysis phase. In what follows, a description is given of the interactions that occur between the user and the MCCSA during each phase of the experiment.

2.3.1 Experiment Definition

During this phase, the user defines all aspects of his experiment to the MCCSA through the use of a mechanism called a form. A form is a collection of data that describes the workings of a test bed device, the layout of a screen to be displayed during the running of the experiment, the actions to be taken in response to conditions on the test bed, or the totality of forms and other data that comprise an experiment. A master template for each form is created initially and modified thereafter by changing the underlying structure of a

THE ELEMENTS OF
SCIENCE

TABLE 2 (Cont'd)

REQUIREMENTS SUMMARY

TEST-BED REQUIREMENTS	OPERATIONS REQUIREMENTS	SOFTWARE REQUIREMENTS
16 STATUS LINES PER SLAVE ON HIGH-SPEED INTERFACE	NO CHANNEL > 50 NMS	MAX BURST ERROR DURATION 1.1 SECONDS AT 50 NMS
8 STATUS/8 CONTROL LINES PER SLAVE	2 x 10 ⁹ BIT EPISODES RECORDING	1.1 SECONDS AT 50 NMS
CONTROL/STATUS LINES AVAILABLE ON EACH EXTERNAL INTERFACE	1 WRITE ON-LINE STORAGE IN MASTER RATE 500 KBPS/SLAVE	32 LCOPS, 31 SLAVES AND MASTER
1 HIGH-SPEED (50 NMS) DUAL EX) PER SLAVE DATA CHANNEL (FULL)	1 WRITE ON-LINE STORAGE IN SLAVE	9.5 KEYTE ON-LINE STORAGE IN SLAVE
DATA CHANNELS AVAILABLE ON EACH EXTERNAL INTERFACE	5 SECOND RESPONSE	256 CONTROL EVENTS
DATA CHANNELS AVAILABLE ON EACH EXTERNAL INTERFACE	10 MS CONTROL/STATUS REPORTING INTERVAL	10 MS CONTROL/STATUS REPORTING INTERVAL
DATA CHANNELS AVAILABLE ON EACH EXTERNAL INTERFACE	ACCELERATE INPUT DATA RATE < 50 KBPS/SLAVE	ACCELERATE INPUT DATA RATE < 50 KBPS/SLAVE
DATA CHANNELS AVAILABLE ON EACH EXTERNAL INTERFACE	< 5 NSEC WITH FULL REPORTING	< 5 NSEC WITH FULL REPORTING
PERFORMANCE	5 SECOND CLOSURE UPDATE	5 SECOND CLOSURE UPDATE
PERFORMANCE	> 1.0 NMS ON LCOPS (CORRUPTED)	> 1.0 NMS ON LCOPS
PERFORMANCE	GATHER 50 MBPS DATA	GATHER 50 MBPS DATA
PERFORMANCE	SUPPLY 50 MBPS DATA (PULSE)	SUPPLY 50 MBPS DATA (PULSE)
PERFORMANCE	10 MS INTERROGATION INTERVAL	10 MS INTERROGATION INTERVAL

$$\text{* ERROR RATE} = (\text{ERROR PROBABILITY}) \times (\text{DATA RATE})$$

**EXPERIMENT DURATION LIMITED
ONLY BY RESOURCES**

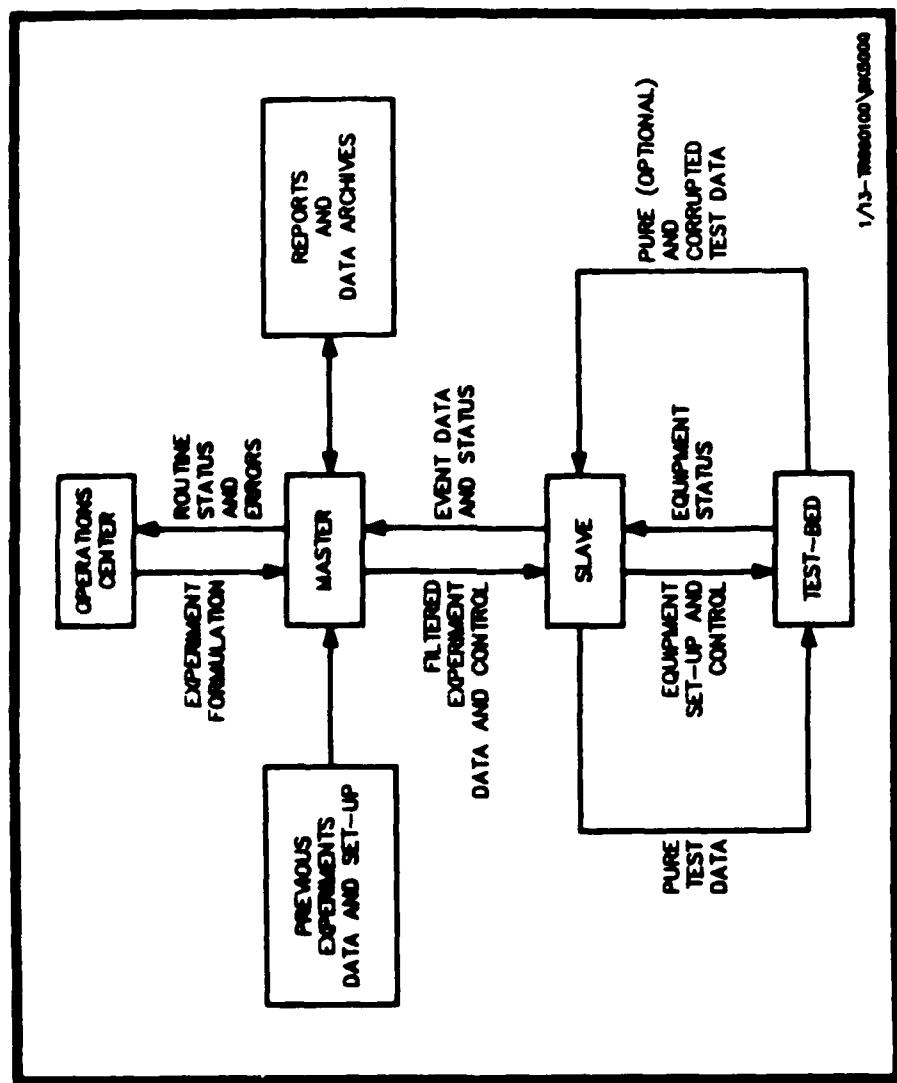


Figure 4. High Level Operations.

copy of the master template or by changing the value of a substitution parameter. Substitution parameters can be as simple as the value of a device setting (data rate, power output, etc.) and as complex as a conditional expression designed to control actions on the test bed. Most of the time, form modification consists of changing a substitution parameter.

A form is usually created by one experimenter and exists for the life of the MCCSA, being borrowed and transformed as different experiments are done. For example, one user may establish a form that describes a particular modem, use it in an experiment, and pass the form onto the next user for a new experiment. The later users may only have to change a few substitution parameters to adapt the modem for use in their work. As time progresses, a library of forms will develop that make future experiments easy to do. The most portable forms will undoubtedly be those that describe equipment or displays, followed by forms governing the actions carried out during the experiment, followed by forms describing the experiment itself.

As each substitution parameter or field is completed, it is given a quick syntax check to verify that the substituted value conforms to the predefined format of the field. That is, if a substitution parameter is defined to be an integer between 1 and 10, the user's input is checked against this predefined format and the user alerted if the input is erroneous. All forms that constitute an experiment need not be completed in one session. Partially completed forms may be stored for later completion. When all forms are complete, they are passed onto the system compiler to create the experiment portion of the MCCSA's real-time executive. Any errors discovered during compilation are reported to the user; otherwise the compiled experiment may be linked into the real-time executive in preparation for the actual run. During the linkage phase of the pre-experiment many routines that control the test bed are bound together to form the real-time executive. These pieces include all of the low level device drivers implicitly required by the user to control the various PC interfaces, special MCCSA programs to perform basic calculations such as bit error rate, the firmware that is to reside in the HSSDC custom component, etc. By placing the majority of the system control in the software, the MCCSA becomes an extremely flexible tool for analyzing complex communications experiments.

2.3.2 Experiment Phase

During the Experiment Phase, the MCCSA carries out the task that was described by the user in the Experiment Definition Phase. This phase begins by loading an MCCSA slave computer with a copy of the real-time executive. Once the load has been accomplished, the MCCSA waits for an instruction from the operator to begin. At this point the MCCSA begins to exert control over the test bed, carrying out the instructions given it by the experimenter. The MCCSA will feed the data pattern to the test bed if required, collect data from the test bed, monitor events happening on the test bed, and responds to conditions on the test bed automatically.

The MCCSA affords three levels of user interaction that enable the user to run or debug an experiment. The first of these operational modes restricts the amount of user intervention, allowing the user to start, stop, or continue an experiment. This mode is primarily designed for automatic, unattended experiments and will be the mode used most often during the running of an actual experiment. The second mode of operation gives the user a limited intervention privilege, allowing him to query the MCCSA about the values of variables, settings of equipment, status of the machine, etc. The third operational mode gives the experimenter the greatest control over the test bed. In this mode the MCCSA stops every time an event occurs and asks whether it should carry out the action associated with the event. The user may then examine the state of the MCCSA, or continue the experiment by telling the MCCSA how to proceed. This last mode is provided to give the user a powerful debugging tool to verify that the operations of the MCCSA are as the user intended.

The MCCSA has the ability to report the real-time status of the test bed and the perceived state of the data stream. The user may customize the screen content during the experiment definition to display the instantaneous values of different variables. In this way the user can observe the near real-time values of such quantities as the bit error rate, burst error rate, monitoring equipment reports, etc. This can also aid in debugging experiments where problems arise only as a result of the real-time operation of test bed

equipment. This feature can also highlight unexpected events and help the experimenter in the analysis of the communication equipment and links under test.

During the experiment different events may occur on the test bed that warrant observation or demand some action. The MCCSA provides the user with up to 16 ports where one-bit signals can be given to the MCCSA to elicit some action. Other conditions on the test bed as well as events internal to the MCCSA may also cause some actions. These conditions and their associated actions can all be specified during the Experiment Definition Phase. Any reasonable action may be taken in response to test bed conditions including the reading of monitoring equipment, the resetting of various, controllable test bed parameters (power level, data rate, etc.), the recommanding of test bed equipment (resynchronize, reacquire, etc.), or the calculation of other parameters based on current readings.

Some internal events will have automatic responses. These include alert conditions present in the MCCSA (out of storage, loss of HSSDC synchronization, etc.) as well as a timer that can be used to perform periodic actions (read monitoring equipment, calculate parameters such as bit error rate, etc.).

When events occur, an event token is created in the MCCSA that contains information about the nature of the event and the number of the bit that was last received. This later feature gives the user the ability to mark the time at which events occur without the need for an extremely accurate, system wide clock. The time at which events occur is accurate to within one bit after processing bias has been removed. This method of timekeeping should provide the necessary accuracy for any experiment.

2.3.3 Experiment Analyses

After the experiment is concluded, all of the data that has been collected during the experiment is put onto magnetic media for later processing. The MCCSA is supplied with a statistics package that lets the user calculate such

elementary statistics as bit error rate or burst statistics, and provides a time history of the experiment showing how the statistics correlate with the events that occurred on the test bed. This package also gives the user the ability to filter the data so that only certain records are analyzed by the statistical package. This can be useful when calculating statistics dependent on the existence of test bed events or conditions.

The files in which the data is stored can also be processed by a user's custom, analysis programs. The filtering provided by the statistics package should free the user from having to perform extensive, data reduction chores and allow him to produce results quickly and easily.

After the experiment phase concludes, or after the user finishes an analysis session, the data files generated are catalogued so that they may be easily retrieved. The MCCSA helps the experimenter keep track of his progress by providing on-line assistance in locating old experiments similar to his own or containing related results data. The information stored with each experiment contains such data as the experiment logistics (who, where, when, etc.), test bed layout, forms used, location of results; and may be searched by key words and phrases. In this way the MCCSA functions not only as a device to run an experiment but also as an experiment management tool as well.

2.4 HARDWARE SUMMARY

This section addresses the hardware that was specified as part of the MCCSA. The two types of hardware components, off-the-shelf and custom-built, are discussed separately.

2.4.1 Off-The-Shelf Hardware

The MCCSA consists of two or more IBM PC ATs (or equivalent) connected together by a Local Area Network (LAN). One AT, called the master, is the computer on which the experiment is developed. This is the computer on which the forms are created, and on which the experiment is compiled and linked into an integral whole. The other computers on the LAN, called slave computers,

have the task of controlling the test bed and reporting status and events to the other slaves on the LAN and to the master. The master consists entirely of relatively inexpensive, commercially available equipment. The majority of the slave hardware is also available commercially but the slave also contains the custom component called the High Speed Serial Data Comparator (HSSDC). This custom component will be discussed at some length in the sequel.

Table 3 is a list of all the off-the-shelf hardware required to build the full MCCSA, (lesser versions can be created from a subset of Table 3).

The master computer contains a fast processor (an Intel 80286 running at 8 Mhz), a numeric co-processor, a hard disk drive(s), floppy disk drives, and a number of interface cards for connecting printers, plotters, cassette storage, tape drives, and the LAN. The master does not connect directly to the test bed but rather communicates with the test bed through the LAN and the slave computers. The slave computers, similar in architecture to the master computer, contain interfaces that permit connection of various test bed devices. The interfaces that are supported are the current, common, serial and parallel interfaces (RS-232-C, RS-422, RS-449, IEEE-488) found in many commercial PC applications as well as two standard military interfaces (MIL-STD-188 and MIL-STD-1553B). With this complement of interfaces, the MCCSA should be able to talk to any current piece of laboratory or communications equipment.

The slave computers also support analog to digital converter cards enabling the slaves to be attached to analog monitoring devices. In this way, some older types of monitoring equipment may also be attached to the MCCSA. The A/D cards support multiple channels of analog input so that many different monitoring devices may be attached to a single card.

Through these interfaces the MCCSA communicates with the test bed, not only reading test bed status or data, but also sending data and commands onto the test bed. At low data rates this is all that is needed to conduct an experiment. It is only when the data rate of a single channel exceeds about 1.5 Mbps that the custom hardware is called upon to control the high speed channel.

TABLE 3
OFF-THE-SHELF COMPONENTS LIST

- A. 80286 BASED PERSONAL COMPUTER (IBM PC AT OR LOOK ALIKE)
- B. RANDOM ACCESS MEMORY (RAM) BOARDS
- C. FIXED/FLOPPY DISK DRIVE ADAPTOR
- D. FIXED DISK DRIVE(S)
- E. FLOPPY DISKETTE DRIVE(S)
- F. CASSETTE TAPE CONTROLLER
- G. CASSETTE TAPE DRIVE
- H. PRINTER ADAPTER
- I. PRINTER
- J. EXTENDED GRAPHICS ADAPTER (EGA)
OR
HERCULES (OR EQUIVALENT) VIDEO CARD
- K. COLOR/MONOCHROME MONITOR
- L. LAN ADAPTOR
- M. EXPANSION CHASSIS
- N. REEL-TAPE DRIVE CONTROLLER
- O. REEL-TAPE DRIVE
- P. AUXILIARY PRINTER ADAPTER
- Q. HIGH SPEED PRINTER
- R. SERIAL INTERFACE BOARD(S)
- S. ANALOG/DIGITAL, DIGITAL/ANALOG INTERFACE BOARD
- T. IEEE-488 INTERFACE BOARD
- U. MIL-1553-B INTERFACE BOARD
- V. UNINTERRUPTABLE POWER SUPPLY

2.4.2 Custom Hardware

The one piece of custom hardware is the High Speed Serial Data Comparator (HSSDC). Figure 5 shows a functional view of this machine. The job of this device is to buffer the data received on a high speed channel against the PC AT's low speed bus and to compress the received data through the use of an efficient compression algorithm. The HSSDC consists of six parts:

- The Host Interface,
- The Experiment Interface,
- The Compare/Compression Engine,
- The Transfer Buffers,
- The Events Engine, and
- The Data Generator.

Each of these pieces is a combination of hardware, firmware, and software designed to keep a 50 Mbps experiment running until all of the available resources are expended. The microprocessor that controls each of the above six components is an 18 Mhz Motorola 68020 microprocessor. This processor was determined to have the necessary speed to keep up with a 50 Mbps data channel (and associated custom hardware) and was chosen solely on that basis. The 68020 processor also contains an internal cache memory that can store a small program which the processor can execute directly. This saves the processor from having to fetch its instructions from an external memory and so saves additional processor execution cycles. All firmware was developed with this cache memory in mind and so the firmware associated with each HSSDC component is a small number of instructions.

2.4.2.1 The Host Interface. The host interface is a 68020 board that can slot into an IBM PC AT and provides a gateway via dual ported memory to the remainder of the HSSDC custom hardware. It is through this interface that the slave computer can learn about the status of the experiment in progress as well as issue commands to the HSSDC that control its configuration and action.

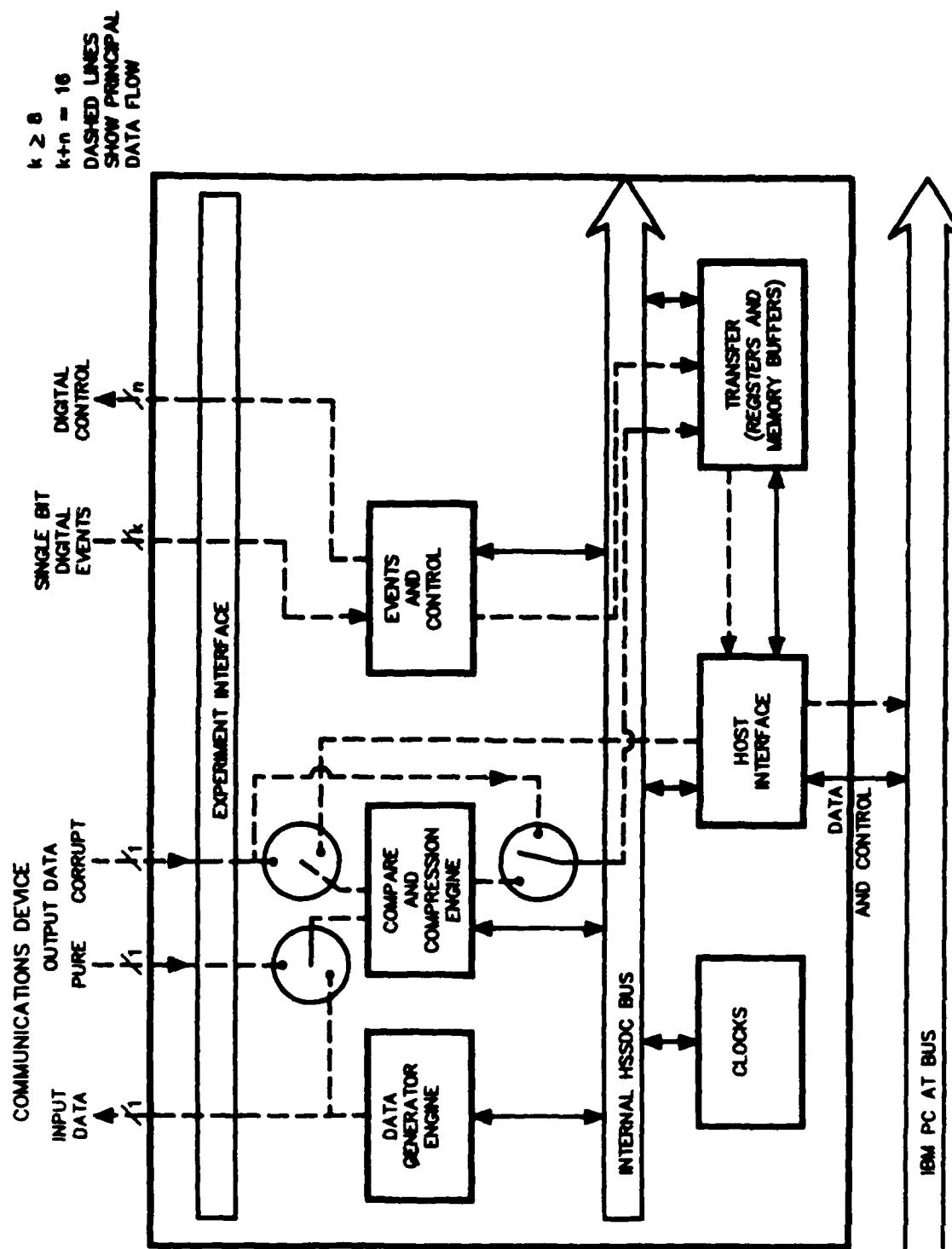


Figure 5. High Speed Serial Data Comparator.

2.4.2.2 The Experiment Interface. The experiment interface is the place where the HSSDC is connected to the test bed. This module contains a fiber optic interface that supplies the test bed with 50 Mbps data and captures two 50 Mbps result streams from the test bed. The experiment interface also contains circuitry to enable the HSSDC to synchronize these two incoming data streams. The two data streams, called the pure stream and the corrupt stream, can originate on the test bed, or the pure stream can be supplied directly by the HSSDC and compared with a corrupt stream extracted from the test bed. It is important to note that both the pure and the corrupt streams should be similar. This means that some alignment of the two streams differ only in those positions where communication's channel errors have occurred.

The synchronization circuit can be programmed to perform two types of synchronization called cyclic and acyclic. Cyclic synchronization is appropriate when the test data stream is a cyclic pattern of bits, and is the preferred mode of operation since synchronization can be achieved with only a modicum of buffering. During cyclic synchronization one word (64 bits) is taken from each of the pure and the corrupt streams and examined for a matching alignment. If no alignment is found the next word is extracted from the corrupt stream and compared with the word after next in the pure stream. By using this technique, two words are drawn from the pure buffer for every one word taken from the corrupt buffer. This prevents the buffers from overflowing and guarantees synchronization when there are no errors and the test data is a periodic bit pattern. After the first alignment is detected the synchronizer microprocessor sets an internal counter to a user defined threshold (the acquisition threshold) and begins to monitor the ensuing comparisons. Every time the alignment is reverified the microprocessor decrements this counter, and each time a misalignment is perceived the counter is incremented. If the counter reaches zero, synchronization is declared. Thereafter, the microprocessor continues to watch the aligned/misaligned condition decrementing or incrementing this counter as before. If the counter ever exceeds a second, preset threshold (the tracking threshold), synchronization loss is declared and the whole process begins again.

Acyclic synchronization is the technique used to align two data streams that are not periodic. The first word of the pure stream is compared with the words received on the corrupt channel until an alignment is found. At this stage, the algorithm continues as in the cyclic case with the ensuing comparisons of both streams driving the microprocessor's acquisition and tracking counters. The important fact to observe here is that the synchronizer must hold the incoming pure stream until the first alignment of the pure and the corrupt stream is found. Naturally, this implies the need for a large buffer on the pure channel to account for the instances where the corrupt stream's arrival is delayed.

If the user is using a satellite channel simulator (or a real satellite channel) there can be as much as .25 second delay through the channel. This translates into a large buffering requirement at high data rates unless cyclic synchronization is performed. Therefore it is recommended that the user send cyclic data patterns over channels where there is the potential for long delays, unless the user can provide his own buffering of the incoming pure data stream.

2.4.2.3 Compare/Compression. This process is accomplished by another custom circuit that tests the two data streams for errors and compresses the information to a reasonable size for later study. The compression also helps match the data rate of the incoming signal with the bus bandwidth of the PC AT. The synchronized streams are given to the compare/compression engine as words (64 bit sections), and a word wide exclusive-or operation produces an error pattern. If this pattern is zero the compression algorithm merely records this fact by incrementing a 64 bit counter. If errors have occurred, the error pattern itself is saved. At the appropriate time, the compression algorithm alerts the transfer engine that data is ready to be forwarded. When this occurs 65 words are given to the transfer engine. These consist of a header word, whose bits describe whether the ensuing words represent error patterns or counts of error-free words, and the error patterns and counts of error-free words themselves.

During periods of error-free transmission, the compression ratio can be extremely large. The worst performance occurs when one or more errors occur approximately every 128 bits. In this case no compression occurs at all and the user is penalized by the presence of the header word. However, if anticipated bit error rate is larger than 1/128 the compare/compression engine can be circumvented and the data streams passed directly to the transfer engine. During a direct transfer, the PC AT's bus bandwidth will be the limiting factor in determining the largest, sustainable test bed data rate. But, it is also possible to discard all of the error pattern data by allowing the internal buffers in the transfer engine to overflow. The MCCSA can be programmed to ignore this data and to concentrate on the statistical counters present in the various engines. Using this tactic the user may run long, high data rate experiments and still capture the relevant statistics of the channel. Events can be recorded as before and correlations done at the experiment's conclusion. The one element that is lost through this technique is the history of error patterns.

2.4.2.4 Transfer Buffers. The HSSDC is provided with 8 MB of storage to hold the output from the compare and compression engine. This is enough storage to allow a 50 Mbps experiment to run with error rates in excess of 1/128 for about one or two seconds before a buffer overflow occurs. If, however, the error rate is less than about 1/256 the HSSDC should be able to provide ample buffering to allow a 50 Mbps experiment to run until the mass storage (i.e. disk drives) are exhausted (a 300 MB disk drive will overflow in about 2 minutes). For experiments with such large error rates it is not recommended that the user run for a long time unless the MCCSA is set to capture statistics only. In this latter mode an experiment could be run for hours and is then limited only by the amount of ancillary data that is being collected.

2.4.2.5 Events Engine. This component of the HSSDC is perhaps the simplest of all of the custom components. This circuitry provides 16 channels (8 of which are bidirectional) through which the MCCSA can read single bit events occurring on the test bed or command single bit ports on test equipment. For example, a modem may only be capable of signaling the loss of synchronization

by turning on an indicator lamp. A light sensor coupled with a level converter could be used to signal the MCCSA, through its events engine, that the modem has lost synchronization. When such an event occurs, the microprocessor on the host interface executes an immediate interrupt and not only determines which event is causing the interrupt, but also records the time that the interrupt occurred. This mechanism creates a time-stamped, event token that is recorded in the event log, used to create the experiment's time history, and analyzed by the real-time executive to determine what (if any) actions to take.

2.4.2.6 Data Generator. The data generator is the custom hardware whose job is to supply the test bed with test data. The data supplied can be either cyclic or acyclic. For cyclic data patterns, the user selects a pattern from a set of prestored patterns or uses a pattern of his own devising. The cyclic pattern is then loaded into the data generator from the PC AT through the host interface. The data generator contains a 128 KB buffer that allows cyclic patterns up to about 130,000 bits to be used. The data generator can also send acyclic patterns to the test bed. To accomplish this the slave PC AT generates a pseudo-random sequence of bits and sends these patterns in blocks of 64 KB (about 65,000 bits) to the data generator through the host interface. The host interface contains status registers that can alert the slave when one of the data generator's data banks is empty. When this happens, another 64 KB block of pseudo-random numbers are generated and sent to the empty data bank. Schemes that are capable of generating long strings of pseudo-random bits are well-known, easy to implement, and fast (See [5]).

2.4.2.7 Firmware. Most of the custom components are driven by microprocessors. The firmware to run these computers may reside, at the user's option, in two places. Provisions have been made to allow the firmware to reside in EPROMs on each of the custom engines. This is done so that each engine will have a default algorithm to run at the time of the experiment. This also frees the user from the task of having to specify an algorithm for each of the microprocessors before running an experiment. However, the EPROMs will always contain a bootstrap loader whose task it is to load the microprocessor code into the 68020 cache memory. This is where custom algorithms can

be loaded into the HSSDC since the bootstrap loader does not have to load the prestored algorithm into the processor but can be directed to load a section of the real-time executive instead (a section that presumably contains the user's custom algorithm).

2.5 SOFTWARE SUMMARY

This section addresses the software that was specified as part of the MCCSA. The two types of software, off-the-shelf and custom written, are described below.

2.5.1 Off-The-Shelf Software

Many of the tasks that must be done by the MCCSA are tasks routinely performed everyday by programmers. These include the compilation of programs, the linking of separately compiled routines into an integral whole, the editing of source code, and the overall running of a computer. These functions are all accomplished by off-the-shelf software packages provided with the MCCSA.

Because of the widespread acceptance and use of the IBM PC, its operating system (PC DOS) has become an operating system for which much software has been written. This includes not only a large number of commercially available software packages but also a vast amount of user supported software (public domain programs, shareware, etc.) For this reason PC DOS was chosen as the operating system for the master MCCSA computer. The slave computers run an operating system called the real-time executive that will be discussed in the section on custom software.

The IBM PC AT contains an Intel 80286 processor. To maintain compatibility with the earlier models in the IBM PC family the PC AT runs the 80286 processor in a mode that emulates, albeit at much higher speed, an Intel 8088 processor -- the original processor in the IBM PC. The major implication of this mode of operation is the maximum amount of Random Access Memory that can be addressed by the processor (1 MB). To overcome this limitation (especially in the slave computers) a Local Area Network operating system was chosen that

not only drives the LAN but also provides access to the full memory space for which the processor was designed (16 MB). This operating system runs PC DOS as a subprocess so that all programs that run under PC DOS should run in this environment as well.

Another area where off-the-shelf products were provided for the MCCSA was in the area of compilers, linkers, and loaders. The C programming language was chosen for its flexibility, power, and currency. Most implementations of C also come packaged with a linker, run-time library, library of common subroutines, and a debugger, while DOS provides the loader function as a built-in facility. No particular vendor's C package was specified since the number and quality of different C packages available at the time of implementation will undoubtedly increase. Information was included, however, that describes the current market offering.

The final area in which off-the-shelf software may be effectively used is in the realm of post-experiment statistics. Many packages were surveyed with respect to the number and type of statistics which they provide. It was also required that the package be capable of working with strictly binary data brought into the program through a disk file, that the package have good graphics and reporting facilities, and that it be executable as a stand alone program under DOS. The ASYST package was chosen as the principal statistical package based on these abilities as well as its ability to create a time correlation history of events. Several other statistical packages are described in one of the software specification's appendices. All are inexpensive and each has its own unique set of features that can free the experimenter from having to design, test, and code large amounts of custom, analysis software.

2.5.2 Custom Software

The MCCSA possesses its own special set of customized software to enable the user to define and run an experiment. The first custom program the user encounters is the context editor. This is a full screen editor that displays the contents of a form (see Section 2.3.1) and allows the user to enter

missing parameters in any order simply by moving the cursor around the screen. The context editor also checks a particular entry against its form definition (size, type, range, etc.) every time a field is altered, and contains an on-line help function to aid the user in understanding the meanings of the various fields on a given form. The contents of the help fields are defined by the individual who first establishes a particular form and serves to reinforce the user's understanding of the device and its corresponding MCCSA form. Through the auspices of the context editor, the user can define the entire experiment's collection of forms (device forms, screen forms, rules forms, and experiment form) and can create new forms at will.

A second piece of custom software is the source-to-source macroprocessor. Each form contains fragments of high-level source code as well as values for the substitution parameters. The source-to-source macroprocessor has the task of inserting the substitution parameter values into the appropriate lines of the language fragments, creating appropriate header and trailer lines for the developing module, and putting the language fragments together in the proper order. Once this task has been accomplished for each form, the resulting high-level source code is given to the system compiler for translation into an intermediate level object code. If errors are discovered at this point, they are reported to the user and control is returned to the context editor so that corrections can be made. If no errors are found, the resulting object code is sent to a linkage editor and combined with the shell of the real-time executive and other programs to create the operating system that controls the slave computer at the time the experiment is run.

This real-time executive is also part of the custom software provided with the MCCSA. Its function is to control the test bed by initializing all of the equipment under its control, by sending the test data onto the test bed, by capturing the results data from the test bed and compressing it when necessary, by controlling other test bed equipment, and by monitoring the status of the experiment. This is accomplished through a set of interface device drivers (usually supplied by the manufacturer of each interface) and by custom software that stores the experiment data on some form of magnetic media, executes actions in response to events on the test bed or in the MCCSA itself, and processes commands received from the LAN.

The final piece of custom software in the MCCSA is the experiment cataloging routines. These are programs that help the experimenter maintain records of his work. The catalog that is produced then becomes part of the system wide records and is available to all other users of the system. In this way, a history of all experiments performed on the MCCSA is maintained. Current users may look through these records to determine if similar experiments were run, and to learn the results of these past experiments. This can be a vital tool in debugging an experiment. Users can also learn the names of the past experimenters or can locate document references to aid their study. In short, this software provides the user community with an experiment maintenance facility and an on-line reference library of past work.

SECTION 3

DESIGN CONSIDERATIONS

This section contains a description of the limitations and the assumptions that were made in the design of the MCCSA. These restrictions were produced as by-products of the attempt to meet the MCCSA design requirements.

3.1 SUMMARY AND ASSUMPTIONS

One of the foremost requirements placed on the MCCSA is the 50 Mbps test bed data rate. While 50 Mbps transmission is well within the grasp of current satellite communications technology, it is on the frontier of computer control. The fastest components available today are just able to keep up with 50 Mbps (undoubtedly by the time the MCCSA is implemented technology will have surpassed this boundary). Hence one assumption made during the design of the MCCSA was that the data rate would not exceed 50 Mbps. There are also other problems that plague the designer at these speeds that are outside the realm of computer control. These include the effects of data distortion (all electronic components and cables begin to act like filters, distorting square wave signals into a composite of sinusoids) and data recovery (it becomes very difficult to recover the data by sampling) to name a few.

Moreover, high error rates at high data rates imply very short duration experiments if an exact error history is required. This may be acceptable for experiments that stop after a large number of bits have been processed (one second of 50 Mbps data may be a large enough sample for some experiments). While computer (PC) technology today is not fast enough to capture a long duration, 50 Mbps data stream can capture 25 Mbps with relative ease. With the development of faster components, 50 Mbps may be easily attainable at the time the MCCSA is built.

However, the direct capture of 50 Mbps data is not desirable because it creates large problems in data management. To store one second of 50 Mbps data requires about 15 of the floppy diskettes in widespread use today. Even

with the advent of the new high density diskettes this number is reduced to only 5. A long experiment then becomes almost impossible to analyze. Even magnetic tape cannot help here. Although one reel of tape can hold about 170 MB of data, the time required to write the tape initially would preclude the running of any high speed experiments (floppy disks suffer from the same limitation also).

To capture the results of a long, 50 Mbps experiment requires some sort of data compression. An effective algorithm was produced, based on earlier work at RADC, and implemented in a combination of hardware and software that tolerates a 50 Mbps input stream and produces an equivalent, slower output stream. The algorithm is based on the assumption that long, sustained periods of high error rate (greater than 1/128) occur infrequently at high data rates. Specifically, it was assumed that the product of the average error rate and the data rate is less than about 1/128. This assumption does not preclude a burst of errors and, in fact, the duration of an error burst (at 50 Mbps) can be as long as about 1.5 seconds before there is a danger of losing the error history. Moreover, if the user is willing to give up the error history during the burst, long bursts can be accommodated and are limited only by the amount of mass storage available.

One of the MCCSA design considerations is that experiments are not to be time limited by the computer components used in the design. This means that hardware components selected for the MCCSA must be physically able to sustain a data throughput of 50 Mbps without starting and stopping the flow of test bed data. Fiber optic interfaces and high speed digital components were selected for the custom hardware to allow data rates as large as 50 Mbps to pass through the MCCSA unthrottled.

Another design consideration is that the duration of an experiment should only be limited by the amount of storage available at the time the experiment is run. This goal was achieved by using fast components and by installing large buffers to hold the incoming data. If properly set, the MCCSA should only stop receiving and processing data when no room remains on the final storage media. Buffers are allowed to overflow and the experiment may then continue by recording statistics only.

3.1.1 Hardware

The great range of data rates that must be accommodated forces the use of a special medium. Fiber optics was chosen because it meets this requirement and provides a low noise path from the actual channel to the HSSDC experiment interface. All of the components needed to construct a fiber optic interface are well-tested and readily available commercially.

To keep up with a data rate of 50 Mbps requires a good mix of hardware and software. It is important not to put the algorithm entirely in hardware, for the design then loses its flexibility. On the other hand, if the algorithm is completely implemented in software the resulting machine will (probably) not be able to process incoming data at 50 Mbps. To achieve flexibility in the design, the MCCSA uses Motorola 68020 microprocessors running at 18 Mhz or better. Because of the speed differences between the microprocessor and the data stream, it is important that the microprocessor never try to stop the hardware intelligently. This is because the time lag between the sensing of an event, the decision concerning what to do, and the issuing of a stop command may be so long that the hardware state present when the event was sensed may be different from the hardware state when stopped. In the design of the HSSDC components, care was taken to insure that the microprocessor only ever has to trigger processes in the hardware, wait for the hardware to finish its job, and read a few status bits. In most cases the microprocessor does not have to wait since the time required to execute the triggering instruction is sufficiently long that by the time the next instruction is executed the hardware has finished its work.

The 68020 microprocessor was also chosen because of its internal cache memory. This is a small amount of random access memory located inside of the microprocessor itself. When instructions are loaded into this memory the microprocessor may execute these instructions without having to perform instruction fetches from the main memory. This affords the processor a tremendous advantage in terms of the speed with which it can execute a task. Moreover, microprocessor instructions that do not cause branching or require main memory access (usually operations on the microprocessor's internal

registers) are executed in parallel with memory access instructions (they are done at the same time the data is being retrieved from memory). This feature along with the processor's good execution speed make it an ideal choice for the HSSDC components.

Another consideration in designing the MCCSA was the speed of the master and slave computers. The current top speed of an IBM PC AT is 8 Mhz. The speed at which IBM PC AT compatibles operate vary from manufacturer to manufacturer, but usually it is possible to find one that operates at speeds in excess of 8 Mhz. In choosing a compatible PC AT consideration must be given to other factors besides performance. These include such intangibles as the perceived worth of the equipment (knowledge gathered by reviews or from word of mouth), the economic health of the manufacturer, the relative ease of obtaining service when the machine malfunctions, the peripherals supported by the compatible, etc. The PC AT running at 8 Mhz is adequate as a master or slave computer in the MCCSA system, and all PC AT peripherals are designed with the IBM system in mind. However, there are other machines that will also work (for example, Compaq and Sperry are well established companies that manufacture equivalent machines) so no recommendation of a specific machine is made in the hardware specification. To make a specific recommendation would date the specification before anything was even built.

3.1.2 Software

Since the cost of training an operator on any piece of equipment can be quite large, and because personnel turnover is present in any organization, the software for the MCCSA is designed so that a minimal amount of effort is required to learn to operate the MCCSA. Moreover, extensive on-line help is provided with all of the custom software where appropriate (the context editor is a notable example).

The design of the software is modular and structured so that creation and maintenance of the custom software can be done with a minimum of difficulty. Moreover, the forms have been designed to move from experiment to experiment so that once a form has been established for one piece of equipment, or for a

particular procedure, it need not be reproduced by later users. The information concerning the library of existing forms is part of the cataloging procedures built into the conclusion of each experiment. In this way a naive user may be able to create an entire experiment out of previously created forms. This aspect of the MCCSA is designed to make the creation of new experiments as painless as possible.

A prime consideration during the design of the MCCSA was to make this machine as versatile a tool as possible. This means making as few assumptions as possible about the test bed equipment and conditions. Initially, the MCCSA will come with forms that describe the complete, usable set of equipment at RADC, and some common types of event processing that should happen during a typical experiment (the calculation of instantaneous bit error rate, for example). Moreover, some general forms will also be provided so that when new equipment arrives it can be fitted into the MCCSA scheme easily. The forms supplied with the MCCSA will not only let experimentation begin immediately but also will provide several examples showing how forms are constructed -- a feature that should aid immensely in the creation of new forms.

SECTION 4

PROBLEMS AND SOLUTIONS

This section chronicles some of the problems encountered during the design of the MCCSA that were produced by the system requirements. Summaries are also given that describe the solutions that were achieved.

4.1 SYNCHRONIZATION

The synchronizer was one of the major challenges of the design of the MCCSA. The problem is to align two data streams so that pure bit n and corrupt bit n are presented at the same time to the circuitry that follows. The solution consists of analyzing all possible alignments of the two streams until a match is found. This is achieved in hardware by scanning 64 bit sections of the inputs until an alignment is produced. Since the transmitted bit pattern usually is longer than 64 bits care must be taken so that synchronization is not declared prematurely. Moreover, errors present in the corrupt stream may also distort the true alignment.

The solutions to these problems were discovered thanks to many suggestions made by individuals at RADC. The main portion of the synchronizer consists of a circuit that sequentially compares a 64 bit portion of the pure stream with each of the 64 alignments of this data with a 128 bit section of the corrupt stream. Each of the 64 match patterns (a 64 bit wide exclusive or) is passed to a set of four Read Only Memories (ROMs) that produce the number of mismatches present in each 16 bit section of the original 64 bits. These values are then given to a Random Access Memory (RAM) that reports a one or a zero depending on the numbers produced from the ROM. The user is free to load the RAM table in any desired manner. One loading of the RAM may produce a criteria that declares a match if k or fewer mismatches are found in 64 bits, but other criteria are possible as well (synchronization based on the parity of the number of matches, matches if forbidden positions, etc.). The synchronizer then counts the number of times the criteria is met and declares synchronization when the count exceeds a prechosen value. Loss of

synchronization is also asserted if a prespecified number of ensuing 64 bit words fail the synchronization criteria.

The ROM and RAM components were chosen because they provide one of the fastest, computerized ways of calculating a function of one variable. Not only is table look-up generally a fast technique, but the RAM table gives the user the ability to easily modify the function being calculated. This works to the user's benefit in the MCCSA since the error rate of the corrupt stream will vary from experiment to experiment. The microprocessor also contains an internal counter to prevent false synchronizations and premature loss of synchronization. This counter also gives the user the ability to run experiments where the error rate of the corrupt stream temporarily increases because of error bursts, deep fades on the channel, etc. The synchronizer will maintain the alignment of the two streams (for a while) even though the streams repeatedly fail the synchronization criteria.

It was also decided to provide two types of synchronization -- cyclic and acyclic synchronization. The former is preferred and should be used whenever an end-to-end test of a channel is being conducted. This form of synchronization allows the synchronizer to work most efficiently and eliminates the need for a large buffer on the pure channel. However, cyclic synchronization is not appropriate in all cases, especially when both the pure and the corrupt stream are taken from test points on the test bed. For example, to determine the effects of attacking the header information in a protocol packet may require that the pure stream be a faithful copy of the packet and that the corrupt stream be a damaged copy. Even if cyclic data is supplied to the channel there is no guarantee that the output of the protocol encoding device is going to be cyclic. This is a case where acyclic synchronization together with adequate pure channel buffering should be used.

4.2 TIME STAMPING

Another problem encountered in the design of the MCCSA was the accurate recording of events during high data rate experiments so that meaningful events histories could be produced at the experiment's conclusion. The pure

data stream acts like a natural clock in any experiment so rather than recording time using a separate clock, which would be acting asynchronously to both the data and events, the pure data stream is sent to a counter that keeps a record of the number of bits processed by the HSSDC. When an external event is sensed by the event engine the first action of the controlling microprocessor is to read the bit count and then the event status register itself. In this way a token of the event is created whose age at creation is a function of the time it took to service the interrupt (a constant since this is the only interrupt to which this processor must respond) plus the time required to read the bit count (again a constant). This bias can be easily determined during implementation of the MCCSA and then removed during the post-experiment analysis. Hence the recording accuracy of external events is quite good.

Internal events (buffer overflow/empty conditions, 10 millisecond clock ticks for performing screen updates) were deemed less time-critical than their external counterparts. The occurrences of these events are also marked with the bit count but because these events must be serviced by multi-tasking microprocessors the time tags of these events will contain errors of varying, small sizes that reflect (for the most part) the interruption priority of the event. The errors seen in these events will probably be accurate to about 64 bits.

4.3 COMPRESSION

The compression algorithm incorporated into the MCCSA is a modified version of a compression algorithm used in the past at RADC. As with most algorithms of this kind, compression is not guaranteed all of the time. The algorithm compares two 64 bit sections of the pure and corrupt stream to create an error pattern. If the error pattern is all zeros then no errors are present, otherwise errors are present. In the case of errors, the pattern is sent on for further processing. If no errors occur, a 64 bit counter is incremented to indicate the number of 64 bit error-free words that have been received since the last error. While this strategy can provide excellent compression when the error rate is low, it can actually cause a slight expansion of the

data stream if the error rate exceeds 1/128 since a header word describing the type of the words that follow always precedes the actual list of error patterns or error-free counts. Thus if one error is received every 128 bits, 64 words containing alternating error patterns and error-free counts (of 1) will be sent on, including a header word of alternating 0 and 1 bits (0 = error pattern), causing an expansion of about 1.5 percent.

Several alternatives are available to manage this problem. First of all, provisions have been made in the MCCSA design to bypass the compare/compress engine. In this case the data is given directly to the transfer engines for storage and the compression overhead is eliminated. This, of course, implies that no compression was done and so analysis of the data may be cumbersome.

Since the compare/compression engine is not microprocessor controlled, it is difficult to change the compression algorithm. Alternate algorithms were considered during the design of the MCCSA but were equally flawed over some region of bit error rate. That is, these algorithms worked well at high error rates but failed in the same or similar ways at moderate or low error rates. Because the choice of compression algorithm is intimately tied to assumptions about the error rate, it was decided to provide a single algorithm for compression that worked well at all but the largest of error rates and, if desired, allow the algorithm or be bypassed, or allow the HSSDC to produce only statistics by discarding the error history.

4.4 HARDWARE CHOICE

The choice of the IBM PC AT (or equivalent) was an easy one indeed. This microcomputer has widespread acceptance in industry and many third-party vendors have developed a plethora of inexpensive, interfaces for this machine. The amount of hardware available for the IBM PC is only surpassed by the amount of software (both commercial and public domain) available for this machine. Since IBM introduced the machine 5 (or so) years ago, many and varied software development environments have been produced to aid both users and developers. Moreover, this line of equipment and its peripherals, are easy to maintain -- either through service contracts available from most

dealers, from third-parties who deal exclusively in the repair of PCs, or by carrying the malfunctioning equipment directly to the repair facility itself. Repair is typically inexpensive and can be accomplished usually in a few days.

4.5 SOFTWARE CHOICE

As with the hardware, the choice of software was also an easy task given the vast offering available for the IBM PC. Since Microsoft's PC DOS has become a de facto standard, this operating system was chosen to run in the MCCSA to provide access to all of the software that has been developed for the IBM PC. However, PC DOS only allows the user access to about 640 KB of memory in any one machine. This was reasonable in the original PC, which could only address 1 MB of storage at best, but is quite restrictive in the newer machines based on the Intel 80286. This processor can address 16 MB of memory and so to exploit this feature without having to discard PC DOS, several other operating systems, which can run PC DOS as a process, were examined. The final choice, Nouvelle's Netware 286, not only solves the PC DOS problem but also is capable of driving the inter PC Local Area Network (LAN) as well.

One other problem area lay in the choice of the statistical package. Because of the binary nature of the output of the HSSDC it is required that the statistical package be capable of working with both binary files and non-binary files alike. A search of the available statistical packages revealed several that could work with these types of files. The ASYST package was chosen because of the large number of other features it offers as well (other statistical tests, types of outputs, etc.).

4.6 CUSTOMIZATION

Even in the custom arena, care was taken to select components (chips and so forth) well within the confines of current technology. The design of the MCCSA was accomplished without specifying the fabrication of custom chips, etc. Custom chips present the implementer with many more problems than would otherwise be encountered and can considerably delay the appearance of the

finished product. The construction of the HSSDC (acquisition of parts, fabrication, testing, etc.) will account for the the majority of its cost. By specifying only off-the-shelf components be used in the HSSDC the cost of the parts can be kept to a minimum.

REFERENCES

- [1] "Multi-Channel Communications System Analyzer, Design Plan," Stanford Telecommunications, Inc., TR860100b, 19 September 1986.
- [2] "Multi-Channel Communications System Analyzer," B1 Hardware Specification, Stanford Telecommunications, Inc., SP860100, 9 February 1987.
- [3] "Software Requirements Specification for the Multi-Channel Communications System Analyzer," Stanford Telecommunications, Inc., SP860102, 4 February 1987.
- [4] "Engineering Drawings and Associated Lists for the Multi-Channel Communications System Analyzer," Stanford Telecommunications, Inc., TR860163, 9 February 1987.
- [5] Knuth, D.E., "The Art of Computer Programming", Volume 2, (Seminumerical Algorithms), Ch 3, Addison-Wesley, 1981.

MISSION of *Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C³I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C³I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.

END

10-87

DTIC